Intel® Itanium® 2 Processor

Specification Update

July 2002

Notice: The Intel® Itanium® 2 processor may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are documented in this specification update.

Document Number: 251141-001

Contents

Revision History

Revision Number	Description	Date
001	Initial release of this document.	July 2002

Preface

This document is an update to the specifications contained in the Affected Documents/Related Documents in the table below. This document is a compilation of device and documentation errata, specification clarifications and changes. It is intended for hardware system manufacturers and software developers of applications, operating systems, or tools.

Affected Documents/Related Documents

Title	Document #
Intel® Itanium® 2 Processor at 1.0 GHz and 900 MHz Datasheet, Rev 1.0	250945
Intel® Itanium® 2 Processor Hardware Developer's Manual, Rev 1.0	251109
Intel® Itanium® Architecture Software Developer's Manual, Volume 1: Application Architecture	245317
Intel® Itanium® Architecture Software Developer's Manual, Volume 2: System Architecture	245318
Intel® Itanium® Architecture Software Developer's Manual, Volume 3: Instruction Set Reference	245319
Intel® Itanium® Architecture Software Developer's Manual Specification Update	251141
Intel® Itanium®2 Processor Reference Manual for Software Development and Optimization	251110
Intel® Itanium® Processor Family System Abstraction Layer Specification	245359

Nomenclature

S-Spec Number is used to identify products. Products are differentiated by their unique characteristics, e.g. core speed, L3 cache size, package types, etc. Care should be taken to read all notes associated with each S-Spec number.

Errata are design defects or errors. Errata may cause the Intel® Itanium® 2 processor's behavior to deviate from published specifications. Hardware and software designed to be used with any given processor must assume that all errata documented for that stepping are present on all devices unless otherwise noted.

Specification Changes are modifications to the current published specifications for the Itanium 2 processor. These changes will be incorporated in the next release of the specifications.

Specification Clarifications describe a specification in greater detail or further highlight a specification's impact to a complex design situation. These clarifications will be incorporated in the next release of the specification.

Documentation Changes include typos, errors, or omissions from the current published specifications. These changes will be incorporated in the next release of the specifications.

General Information

Intel® Itanium® 2 Package Marking

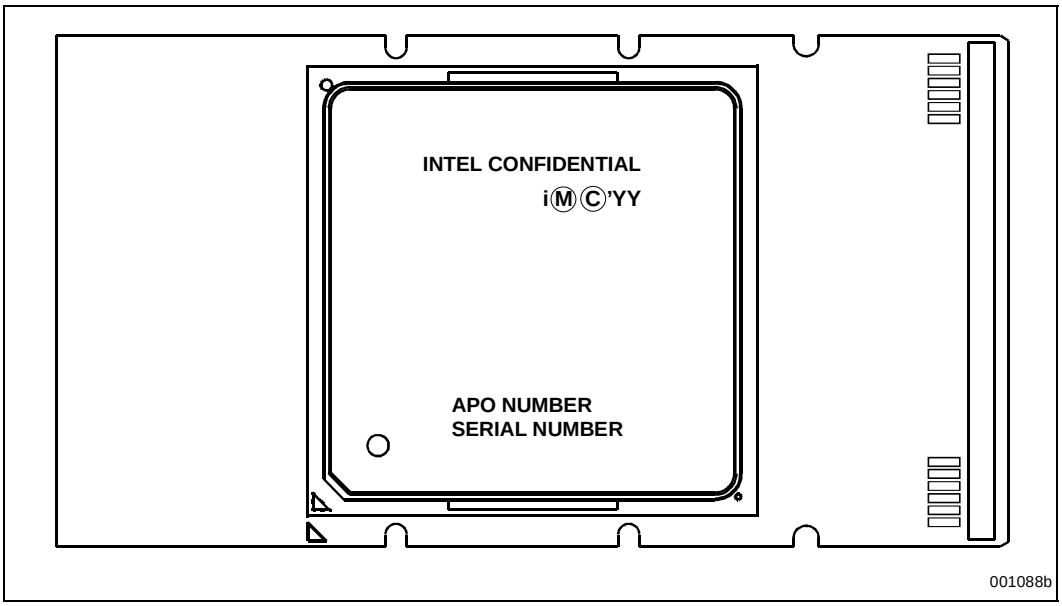
The following section details the processor top-side and bottom-side markings for the Itanium 2 processor and is provided as an identification aid. The processor top-side mark for the product is a laser marking on the Integrated Heat Spreader (IHS).

Processor Top-Side Marking

Figure 1-1 shows an example of the laser marking on the IHS. The processor top-side mark provides the following information:

- INTEL Brand/ INTEL Product
- Legal Mark
- Assembly Process Order (APO) number
- Serial Number

Figure 1-1. Processor Top-Side Marking on IHS

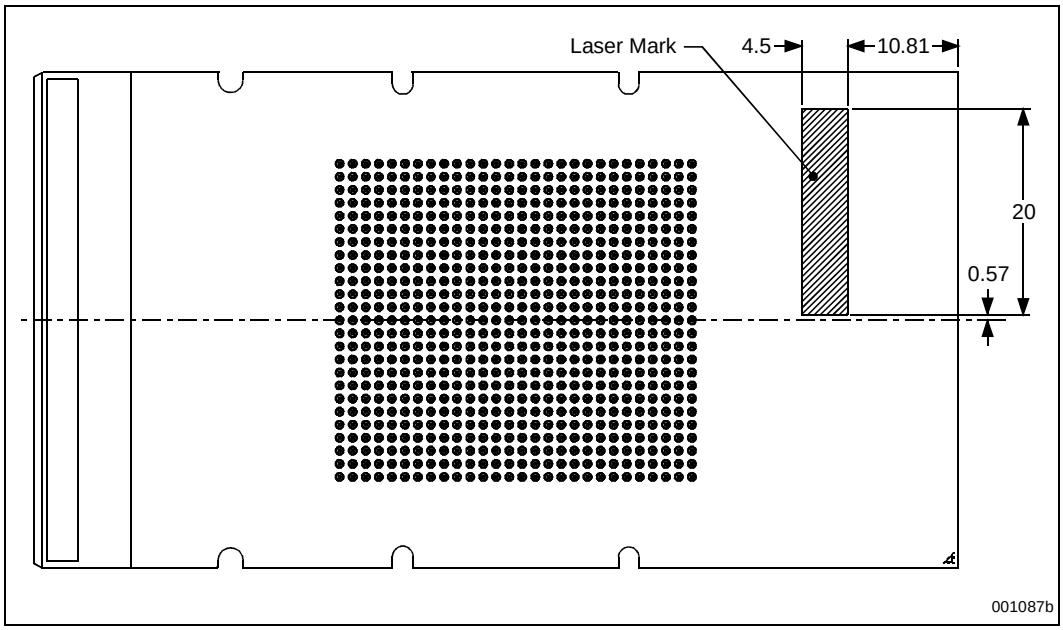


Processor Bottom-Side Marking

The processor bottom-side mark for the product is a laser marking on the pin side of the interposer. Figure 1-2 shows the placement of the laser marking on the pin side of interposer. The processor bottom-side mark provides the following information:

- Product ID
- Finish Process Order (FPO)
- Serial Number
- S-Spec
- Country of origin

Figure 1-2. Processor Bottom-Side Marking Placement on Interposer



Intel® Itanium® 2 Processor Identification and Package Information

S-Spec Number	Processor Stepping	CPUID	Speed (MHz)	L3 Size (Mbytes)	Notes
SL67U	B3	001F000704h	1000/400	1.5	
SL67V	B3	001F000704h	1000/400	3	
SL67W	B3	001F000704h	900/400	1.5	

PAL Version	Processor Stepping	Notes
7.13	B3	

Summary Table of Changes

The following table indicates the errata, specification changes, specification clarifications, or documentation changes which apply to the Itanium 2 processor steppings. Intel may fix some of the errata in a future stepping of the component or in a future release of the Processor Abstraction Layer (PAL), and account for the other outstanding issues through documentation or specification changes as noted. This table uses the notations indicated below.

Codes Used in Summary Table of Changes

Stepping

X:	Errata exists in the stepping or PAL version indicated. Specification Change or Clarification that applies to this stepping.
(No mark) or (Blank box):	This erratum is fixed in listed stepping or specification change does not apply to listed stepping or PAL version.

Page

(Page):	Page location of item in this document.

Status

Doc:	Document change or update will be implemented.
Fix:	This erratum is intended to be fixed in a future step of the component, or in a future release of PAL.
Plan fix:	This erratum may be fixed in a future stepping of the product.
Fixed:	This erratum has been previously fixed.
NoFix:	There are no plans to fix this erratum.

Row

▌	Change bar to left of table row indicates this erratum is either new or modified from the previous version of this document.

Errata Summary Table

Errata

No.	Processor Stepping	PAL Version	Pg.	Status	Errata
	B3	7.13			
1	X		12	NoFix	IA64_INST_RETIRED and IA64_TAGGED_INST_RETIRED does not count predicated off instructions
2	X		12	NoFix	Performance Monitor Interrupt raised when freeze bit is written to Performance Monitoring Counter register
3	X		12	NoFix	Priority agent requests with unit mask of I/O not counted
4	X		12	NoFix	Incorrect fault reporting on move to/from the RNAT or BSPSTORE application registers
5	X		13	NoFix	Power good deassertion affects boundary scan testing
6	X		13	NoFix	IA-32: CPUID instruction returns incorrect L3 cache size
7	X		13	NoFix	Performance Monitoring Event counters may be incorrect when using Instruction Address Range checking in fine mode
8	X		14	NoFix	Possible deadlock condition after ptc.g is issued on two-way system
9	X		14	NoFix	EPC, mov ar.pfs and br.ret instructions may combine to yield incorrect privilege level
10	X		15	NoFix	Removal of WAW hazard may lead to undefined result
11	X		15	NoFix	Unexpected data debug, data access or dirty bit fault taken after rfi instruction
12	X		16	NoFix	Incorrect privilege level may be granted if a failed speculation check precedes a privilege level change
13	X		16	NoFix	Floating-point instructions take a floating-point trap before Unimplemented Instruction Address trap
14		X	16	Fix	PAL_MC_ERROR_INFO does not return an address for certain double bit ECC memory errors
15		X	16	Fix	PAL_CACHE_READ and PAL_CACHE_WRITE return incorrect status for L1I cache access
16		X	17	Fix	Unpredictable behavior if the system is awakened from low power mode by an MCA
17		X	17	Fix	The system may lose an interrupt when SAL_CHECK reads the IVR
18		X	17	Fix	A bus MCA nested within a recoverable or firmware-corrected bus MCA may not be handled correctly
19		X	17	Fix	PAL reset sequence performed after a recovery check may result in incorrect system behavior

Specification Changes

No.	Processor Stepping	PAL Version	Pg.	Status	SPECIFICATION CHANGES
	B3	7.13			
					None for this revision of the Specification Update

Specification Clarifications

No.	Processor Steppings	PAL Version	Pg.	Status	SPECIFICATION CLARIFICATIONS
	B3	7.13			
					None for this revision of the Specification Update

Documentation Changes

No.	Processor Steppings	PAL Version	Pg.	Status	DOCUMENTATION CHANGES
	B3	7.13			
					None for this revision of the specification Update

Errata

1. IA64_INST_RETIRED and IA64_TAGGED_INST_RETIRED does not count predicated off instructions

Problem: The event monitor count for instructions retired (IA64_INST_RETIRED and IA64_TAGGED_INST_RETIRED) does not include the predicated off instructions.

Implication: The IA64_INST_RETIRED/IA64_TAGGED_INST_RETIRED performance monitoring events may report an incorrect count.

Workaround: Add the PREDICATE_SQUASHED_RETIRED event monitor count to the IA64_INST_RETIRED and/or the IA64_TAGGED_INST_RETIRED event monitor count to get the intended results.

Status: For the steppings effected, see the Summary Table of Changes at the beginning of this section.

2. Performance Monitor Interrupt raised when freeze bit is written to Performance Monitoring Counter register

Problem: The Performance Monitor Freeze (PMC[0].fr) bit within the Performance Monitoring Counter (PMC) register is used to stop performance event monitoring. This can be set by software or optionally by an event counter overflow. Due to this erratum, the processor may raise a Performance Monitor Interrupt when the freeze bit is set by software, even when the Performance Monitor Overflow Interrupt (PMC.oi) bit is not enabled and no overflow has occurred.

Implication: The processor may generate a performance monitor interrupt when it is not expected to do so.

Workaround: The interrupt service routine needs to account for the spurious interrupt even if no performance monitor overflow is indicated.

Status: For the steppings effected, see the Summary Table of Changes at the beginning of this section.

3. Priority agent requests with unit mask of I/O not counted

Problem: The system bus allows for the BPRI# signal to be asserted one cycle before an ADS# is driven by the priority agent, provided no BREQ# pins are driven by the processor. Priority agent requests exhibiting this behavior are not counted by the system bus performance monitoring events when using a unit mask of ‘I/O’.

Implication: The system bus performance monitoring events may report an incorrect count in this case.

Workaround: Measure the bus transactions for all bus masters (unit mask= ‘ANY’) and subtract from it the sum of the corresponding bus transactions on each local processor (unit mask= ‘SELF’).

Status: For the steppings affected, see the Summary Table of Changes at the beginning of this section.

4. Incorrect fault reporting on move to/from the RNAT or BSPSTORE application registers

Problem: Incorrect faulting behavior may be experienced under the following conditions:

1. A `mov.imm` (move immediate) to the `ar.rsc` register is executed in the same instruction bundle (or the next bundle with no intervening stop bits) as a mispredicted branch.
2. The mispredicted branch path includes another `mov.imm` to the same `ar.rsc` register, and is within two issue groups or less of the (mispredicted) branch instruction. This instruction is not executed. Also, the value moved to the `rsc.mode` field must be different than the value moved to `rsc.mode` in the `mov.imm` in step 1.

3. The correct branch path is then taken and includes a move to/from the `ar.rnat` or `ar.bspstore` registers, within the first bundle (or second bundle with no intervening stop bit) of the correct branch instruction.

Implication: When the above conditions line up (and there are no stalls or cache misses), the instruction in step 3 (move to/from `ar.rnat` or `ar.bsp`) uses the `rse.mode` value from the `mov.imm` in the mispredicted branch path instead of from instruction in step 1. As a result, there may be incorrect faulting behavior – an illegal opcode fault is missed (if `rse.mode` != 0) or falsely indicated (if `rse.mode` = 0) and may result in inconsistent system behavior. This erratum has only been observed in a system validation environment.

Workaround: Use one of the following workarounds:

1. Use the register form of the move instruction or;
2. Ensure there is a stop bit between any `mov.imm` instruction to/from the `ar.rsc` registers and any subsequent branch instruction or;
3. Ensure that there is a stop bit between a “label” (branch target) and a subsequent move to/from `ar.rnat`/`ar.bspstore`.

Status: For the steppings affected, see the Summary Table of Changes at the beginning of this section.

5. Power good deassertion affects boundary scan testing

Problem: Deassertion of the PWRGOOD signal during Boundary Scan testing prevents the correct operation of the sampling functionality in the EXTEST and SAMPLE/PRELOAD JTAG commands.

Implication: As a result of this erratum the boundary scan chain function is disabled and will stop shifting data when the PWRGOOD signal is low.

Workaround: Keep the PWRGOOD signal asserted during boundary scan testing.

Status: For the steppings affected, see the Summary Table of Changes at the beginning of this section.

6. IA-32: CPUID instruction returns incorrect L3 cache size

Problem: The IA-32 CPUID instruction will always report the L3 cache size as 3 MB even for processors with a cache size of 1.5 MB.

Implication: IA-32 applications using the IA-32: CPUID instruction cannot rely on the cache size reported by this instruction. Native Itanium applications are not affected by this erratum and can access this information via the processor CPUID registers.

Workaround: Within the Linux Operating System environment, the ‘/proc/cpuinfo’ file contains this information. Within the Microsoft Operating System environment this information is available through Windows API calls.

Status: For the steppings affected, see the Summary Table of Changes at the beginning of this section.

7. Performance Monitoring Event counters may be incorrect when using Instruction Address Range checking in fine mode

Problem: For Performance Monitoring Events that use Instruction Address Range Matching set to ‘Fine Mode’ (PMC: 14, bit 13 = 1), the address matching capability will be inconsistent and may yield incorrect results.

Implication: Due to this erratum the results of an event counter while using ‘Fine Mode’ may not be correct.

Workaround: Use normal mode when using Instruction Address Range Checking.

Status: For the steppings affected, see the Summary Table of Changes at the beginning of this section.

8. Possible deadlock condition after ptc.g is issued on two-way system

Problem: In a two processor system, a `ptc.g` instruction is issued on processor A. The execution of the `ptc.g` on processor A, blocks the completion of a semaphore upon which processor B is waiting to become available. Concurrently processor B is issuing a long series of loads and stores with one or more instructions being retried or involves system memory access before being retired. Processor B's L2 cache entry queue, denoted as OzQ, is full and does not allow the `ptc.g` operation from processor A, entry into the L2 OzQ for completion. The `ptc.g` request will be presented again in three clock cycles. If processor B continues to execute a code sequence such that the L2 cache OzQ entries continue to be taken by other load/stores, then the `ptc.g` operation must continue to wait.

Implication: Due to this erratum, the system may deadlock while waiting for the `ptc.g` to be completed. Any break in, or completion of the code loop on processor B, including system interrupts, that allows the `ptc.g` operation to enter the L2 cache OzQ on processor B will be enough to release the deadlock condition. Additional processors will also change the time cycle necessary for this event to occur. This issue has only been observed during Random Instruction Testing in a system validation environment.

Workaround: None identified at this time.

Status: For the steppings affected, see the Summary Table of Changes at the beginning of this section.

9. EPC, mov ar.pfs and br.ret instructions may combine to yield incorrect privilege level

Problem: Due to certain internal timing and microarchitectural conditions, operating system (OS) calls that return to user space from privilege code promote pages using a `br.ret` instruction, may not have the expected privilege level.

Using the following code sequence as an example:

```
<change of privilege level>   //epc on promote page; or br.ret
mov ar.pfs, [value];          //new pfs value has ppl < cpl
br.ret;;
```

In this case the `br.ret` is specified to not change the privilege level (`pl`) since the `br.ret` is asking to promote privilege to a numerically lower level. Current processor steppings may change current privilege level (`cpl`) to the `pl` at the beginning of the <change of privilege level>.

Implication: This erratum would result in having the `cpl` demoted and the user space application may not receive the correct privilege level. Privilege code promote page usage is limited and controlled by the OS. This issue has only been observed during random instruction testing in a system validation environment.

Workaround: Use one of the following workarounds:

1. Use an return from interrupt (`rfi`) instruction instead of `br.ret` to return from privilege code promote pages.
2. Insert a useless call-to-next bundle in all paths leading to a demoting `br.ret`.
3. PAL release 7.01 and above, have a workaround for this issue and it is enabled by default. The OS may implement one of the previous workarounds or a check mechanism, such that this PAL workaround can be disabled. Please review the PAL Release notes for details on the implementation of this workaround.

Status: For the steppings affected, see the Summary Table of Changes at the beginning of this section.

10. Removal of WAW hazard may lead to undefined result

Problem: Due to internal conditions an allowed WAW dependency may become a WAW hazard under the following circumstances:

- A move to the AR.PFS register is followed by a BR.CALL and both are executed in the same issue group, or
- A move to the AR.EC register is followed by a BR.RET and both are executed in the same issue group.

These combinations of instructions are legal WAW memory dependencies if one of the operations is predicated off. If preceding instructions (as indicated above) combine to change the predication on the BR.CALL or BR.RET from predicated true to predicated false, the processor may mistakenly decide the WAW hazard is still present and fail to recognize that the WAW has been removed which may result in an undefined value for ar.pfs or ar.ec.

The following code sequence demonstrates this issue:

```
p15 = 1;
;;
mov ar.pfs = R[x];
ld.c R[y] = [m]; //causes R[y] to be reloaded.

cmp.eq p15, p16 = R[y], R0;
(p15) br.call;
```

The RAW dependencies on ld.c to cmp and cmp to branch are legal. When the processor executes the issue group, the WAW hazard is present and the PFS results are undefined. If the ld.c misses the ALAT, the cmp to branch will be re-executed, the new result of the ld.c causes the p15 value to change to false and thus eliminate the WAW. Then the processor may fail to recognize that the WAW has been removed.

Implication: An application may hang or signal an exception fault under these circumstances. The affected code sequence is not known by Intel to be generated in any current compiled code or exist in any current OS.

Workaround: Separate the predicate producing instruction from its consumer with a stop (as recommended in the *Intel® Itanium® Architecture Software Developer's Manual*, Volume 1: Application Architecture, page 1:157) or change the predication sequence to assure mutually exclusive predication of the instructions in the WAW dependency.

Status: For the steppings affected, see the Summary Table of Changes at the beginning of this section.

11. Unexpected data debug, data access or dirty bit fault taken after rfi instruction

Problem: A fault may be taken after a return from interruption (rfi) instruction has been executed under the following circumstances. The *IPSR.da* or *IPSR.dd* bits are set to disable data debug/data access/dirty bit faults for the first Itanium processor system environment restore instruction. This is followed by an rfi instruction. The rfi instruction is followed by additional instructions that generate RSE activity (alloc, flushrs, br.ret). The processor will see the RSE activity as valid Itanium system instructions and clear the *ipsr.da/dd* bits and this may result in an unexpected data debug, data access or dirty bit fault at the target of the rfi.

Implication: Due to this erratum an unexpected fault may be generated after an rfi instruction has been executed. This may slow the transition of the system into the Itanium system environment and log un-necessary errors.

Workaround: Separate the rfi from the RSE generating instruction by four issue groups of nop instructions.

Status: For the steppings affected, see the Summary Table of Changes at the beginning of this section.

12. Incorrect privilege level may be granted if a failed speculation check precedes a privilege level change

Problem: A failed speculation check instruction (`chk.s/chk.a/fchkf`) that is followed by a privilege change operation may result with the incorrect privilege level for instructions in the issue group of the privilege level change and beyond. The privilege changing instruction must occur within two clock cycles of the failed speculation check.

Implication: As a result of this erratum, the speculation check recovery code and subsequent instructions may have an incorrect privilege level.

Workaround: Do not use speculation near privilege changing instructions. The workaround for this erratum is to escalate failed speculation checks (speculation check re-steers) to the OS for recovery. This workaround is included in PAL version 7.01 and above.

Status: For the steppings affected, see the Summary Table of Changes at the beginning of this section.

13. Floating-point instructions take a floating-point trap before Unimplemented Instruction Address trap

Problem: A floating-point instruction that causes a floating-point trap and is the last instruction at the top of the physical address space should flag an Unimplemented Address trap before the floating-point trap.

Implication: The correct trap is flagged but only after the floating-point trap is taken first.

Workaround: None available at this time.

Status: For the steppings affected, see the Summary Table of Changes at the beginning of this section.

14. PAL_MC_ERROR_INFO does not return an address for certain double bit ECC memory errors

Problem: PAL_MC_ERROR_INFO will report the address for the source of a double bit ECC memory error. However, under the conditions that the data with a 2xECC error was prefetched to the L2 cache and later filled into the L1 cache, the source address will not be available.

Implication: PAL_MC_ERROR_INFO will not be able to report the address of a double bit ECC error in this case. Double bit errors that are consumed in this scenario will be not be recoverable.

Workaround: None at this time.

Status: For the steppings affected, see the Summary Table of Changes at the beginning of this section.

15. PAL_CACHE_READ and PAL_CACHE_WRITE return incorrect status for L1I cache access

Problem: The PAL_CACHE_READ and PAL_CACHE_WRITE procedures should return a status value of '–7' (which indicates this operation is not supported for this *cache_type* and *level*) when attempting to read or write to/from the L1I (instruction) and L1D (data) cache. When these procedures attempt to access the L1I cache an incorrect status value will be returned.

Implication: Due to this erratum, using these PAL procedures to access the L1I cache will result in the return of an incorrect status value, implying that the L1I cache is readable/writeable by these PAL procedure calls.

Workaround: Do not use these PAL procedures to access the L1D and L1I caches.

Status: For the steppings affected, see the Summary Table of Changes at the beginning of this section.

16. Unpredictable behavior if the system is awakened from low power mode by an MCA

Problem: If the system is in low power mode and an MCA, BERR# or BINIT# is signaled, the PALE_CHECK handler will be called to process the error condition. However, PALE_CHECK does not disable low power mode so that it can continue execution. As soon as PALE_CHECK attempts to drain the processor queues, the system may re-enter low power mode. This may cause incomplete handling of the error event and potentially, intermittent continuation of the same event during later signaled BINIT# events.

Implication: The processor can appear to be trapped in low power mode and/or system behavior may be unpredictable.

Workaround: Do not use low power mode or call the following PAL procedures: PAL_HALT, PAL_HALT_LIGHT or PAL_HALT_LIGHT_SPECIAL.

Status: For the steppings affected, see the Summary Table of Changes at the beginning of this section.

17. The system may lose an interrupt when SAL_CHECK reads the IVR

Problem: The PAL_REGISTER_INFO procedure returns an incorrect value to indicate that reading the Interrupt Vector Register (IVR), CR65 (Configuration Register 65) has no side effects. Based on this incorrect return value, when SAL_CHECK reads the IVR while saving system state data to NVRAM, a pending interrupt may be allowed to proceed before the current process has been completed.

Implication: The SAL_CHECK procedure relies on the return values of PAL_REGISTER_INFO to know which ARs and CRs are safe to read and save off. Due to this erratum, the SAL_CHECK reads the IVR, and consequently causes the corresponding bit in the IRR to be cleared and the ISR to change. The results of the interrupt routine currently being executed may be lost.

Workaround: After calling PAL_REGISTER_INFO with *info_request* = 3, SAL can force the correct return value for CR65 by setting bit 1 of *reg_info_2* to a value of one.

Status: For the steppings affected, see the Summary Table of Changes at the beginning of this section.

18. A bus MCA nested within a recoverable or firmware-corrected bus MCA may not be handled correctly

Problem: During the processing of a non-fatal bus MCA, if a second bus MCA is received the second MCA may be missed.

Implication: A bus MCA received in this scenario may be missed and result in unpredictable system behavior. If the first MCA is fatal, system behavior remains correct.

Workaround: None at this time.

Status: For the steppings affected, see the Summary Table of Changes at the beginning of this section.

19. PAL reset sequence performed after a recovery check may result in incorrect system behavior

Problem: The PAL early self-test sequence performed after a recovery check may not properly serialize outstanding memory transactions.

Implication: As a result of this erratum, memory transactions that are outstanding at the point of transition from the recovery check handler to PAL may cause a deadlock condition and possibly hang the processor.

Workaround: SAL can call the PAL_MC_DRAIN procedure before returning to PAL from Recovery Check to ensure that outstanding transactions have completed.

Status: For the steppings affected, see the Summary Table of Changes at the beginning of this section.

Specification Changes

There are no Specification Changes for this revision of the *Intel® Itanium® 2 Processor Specification Update.*

Specification Clarifications

There are no Specification Clarifications for this revision of the *Intel® Itanium® 2 Processor Specification Update.*

Documentation Change

There are no Documentation Changes for this revision of the *Intel® Itanium® 2 Processor Specification Update.*